

# ADT PLM

## Programmer's Learning Machine

Matthieu Nicolas

IJD Seminar, 2016-02-02

## 1 Presentation of PLM

- Purposes
- Demo
- About PLM
- Architecture

## 2 Assessment of user's code

- Challenges
- Extraction of the execution component
- Docker

## 3 Result

## 4 Next steps

## 1 Presentation of PLM

- Purposes
- Demo
- About PLM
- Architecture

## 2 Assessment of user's code

- Challenges
- Extraction of the execution component
- Docker

## 3 Result

## 4 Next steps

# Presentation of PLM

## Purposes

- Application to learn programming.

# Presentation of PLM

## Purposes

- Application to learn programming.
- Allows students to progress at their own speed...

# Presentation of PLM

## Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.

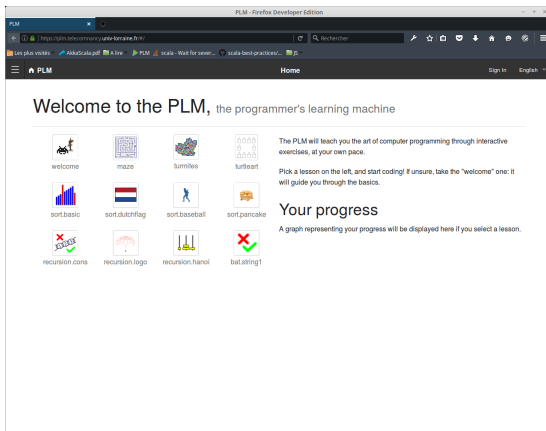
# Presentation of PLM

## Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.
- Used at TELECOM Nancy since 2008.

# Presentation of PLM

## Quick demo



- Available at <https://plm.telecomnancy.univ-lorraine.fr>



# Presentation of PLM

12 lessons, 200 exercises

PLM - Mozilla Firefox

https://plm.univ-lorraine.fr/la/lessons/maze/maze-wallfollower/wallfollowerMaze

PLM

Maze / WallFollowerMaze

Se connecter Java Français

World Objective



Résultats de l'exécution

Éditeur de code:

```
public void run() {  
    // ...  
}
```

Réinitialiser API

Cas de tests:

Another labyrinth Labyrinth

# Presentation of PLM

12 lessons, 200 exercises

PLM - Mozilla Firefox

https://plm.univ-lorraine.fr/.../recursion/hanoi/hanoi.lessons/hanoi.Hanoi

PLM Recursion.hanoi / HanoiBoard

World Objective

0 Move



Résultats de l'exécution

Cas de tests:

solve(0,1,2) solve(1,2,0) solve(2,0,1)

Éditeur de code:

```
public void hanoi(int height, int src, int other, int dest) {  
    ...  
}
```

# Presentation of PLM

12 lessons, 200 exercises

PLM - Mozilla Firefox

https://plm.univ-lorraine.fr/ta/lessons/recursion/logo/recursion/logo/tree.Tre

Les plus visités Linux Mint Community Forums Blog News NinjaMock - free tool... Spotify Web Player Wau The Most Amaz... Coursera P2P

PLM Recursion.logo / Tree Se connecter Java Français

World Objective



Éditeur de code: Réinitialiser API

```
public void tree(int steps, double length, double angle, double shrink) {  
    // ...  
}
```

Résultats de l'exécution

Cas de tests:

tree(10,100,90,0.68)	▶	tree(10,80,45,0.7)	▶
tree(7,75,15,0.8)	▶	tree(7,75,30,0.8)	▶

# Presentation of PLM

Languages and programming languages

- Available languages:
  - English
  - French
  - Brazilian Portuguese
- Supported programming languages:



# Presentation of PLM

## Evolution of the project

- Formerly a fat client
  - Written in Java

# Presentation of PLM

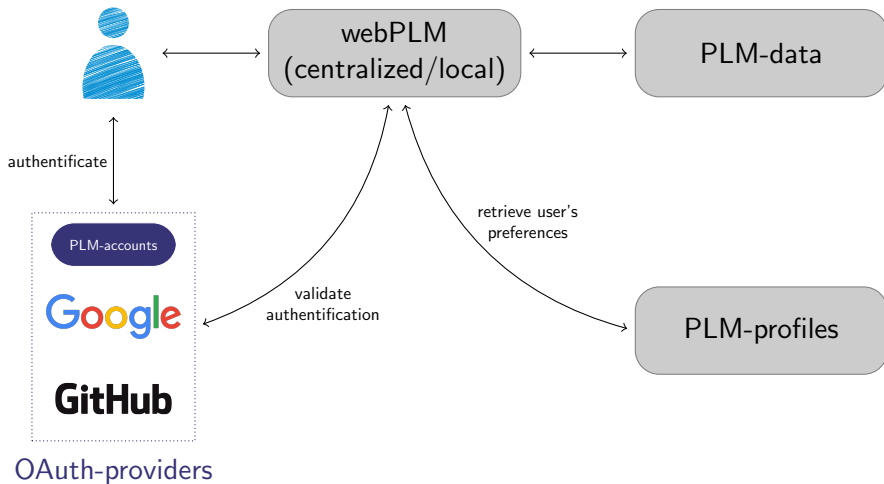
## Evolution of the project

- Formerly a fat client
  - Written in Java
- Switch to a web application
  - Headless version of PLM: PLM-engine
  - Server implemented in Scala using *PlayFramework*
  - User interface written in Javascript using *AngularJS* and *Foundation*



# Presentation of PLM

## Application's architecture



## 1 Presentation of PLM

- Purposes
- Demo
- About PLM
- Architecture

## 2 Assessment of user's code

- Challenges
- Extraction of the execution component
- Docker

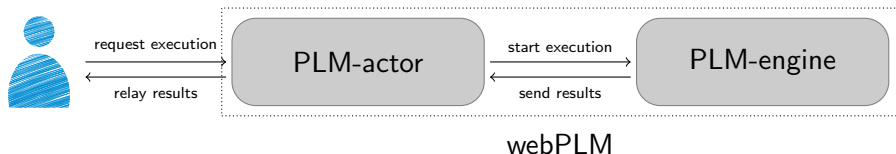
## 3 Result

## 4 Next steps



# Assessment of user's code

## Execution components



# Assessment of user's code

## Limits

- Run on the same machine, same JVM

# Assessment of user's code

## Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops

# Assessment of user's code

## Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation

# Assessment of user's code

## Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation
- And from *System.exit(whatever)*?

# Assessment of user's code

## Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation
- And from *System.exit(whatever)*?
- Scalability issues

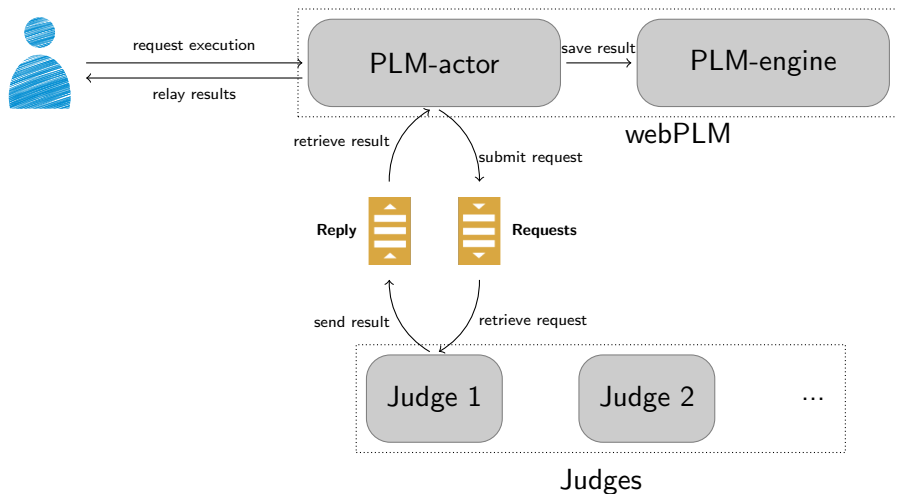
# Assessment of user's code

## Chosen solution

- Delegate the execution to workers
  - Called *Judges* in the litterature
  - Use PLM-engine as well
  - Execute user's code and send back result to webPLM

# Assessment of user's code

## Architecture with judges





# Assessment of user's code

## Implementation

- Distribute workload using message queues
  - One queue for requests
  - One queue per result

# Assessment of user's code

## Implementation

- Distribute workload using message queues
  - One queue for requests
  - One queue per result
- *Let it crash* strategy
  - Prevent obvious issues with a security manager
  - Handle timeout and crash

# Assessment of user's code

## Pros and cons

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements

# Assessment of user's code

## Pros and cons

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements
- Cons:
  - Make sure to use the right version of PLM
  - Need to deploy them easily
  - Should restart them after each execution
  - Have to restrict their resources usage

# Assessment of user's code

## Docker

- Lightweight virtualization tool
- Build image of your application
- Run containers based on images



# Assessment of user's code

## Example of Dockerfile

- Dockerfiles describe how to set up the application

```
Dockerfile
1 FROM node:latest
2
3 MAINTAINER Matthieu Nicolas, matthieu.nicolas@inria.fr
4
5 WORKDIR /app
6
7 # Required for PhantomJS
8 RUN apt-get install -y bzip2
9
10 # Install Mean.JS Prerequisites
11 RUN npm install -g grunt-cli
12 RUN npm install -g bower
13
14 # Install Mean.JS packages
15 ADD package.json /app/package.json
16 RUN npm install
17
18 # Manually trigger bower. Why doesnt this work via npm install?
19 ADD .bowerrc /app/.bowerrc
20 ADD bower.json /app/bower.json
21 RUN bower install --config.interactive=false --allow-root
22
23 # Make everything available for start
24 ADD . /app
25
26 ENV NODE_ENV production
27
28 # Port 3000 for server
29 # Port 35729 for livereload
30 EXPOSE 3000 35729
31 CMD ["grunt"]
```

- Run *docker build -t tag /path/to/Dockerfile* to build the image
- Start containers with *docker run tag*

```
1 FROM node:latest~
2 ~
3 MAINTAINER Matthieu Nicolas, matthieu.nicolas@inria.fr~
4 ~
5 WORKDIR /app~
6 ~
7 # Required for PhantomJS~
8 RUN apt-get install -y bzip2~
9 ~
10 # Install Mean.JS Prerequisites~
11 RUN npm install -g grunt-cli~
12 RUN npm install -g bower~
13 ~
14 # Install Mean.JS packages~
15 ADD package.json /app/package.json~
16 RUN npm install~
17 ~
18 # Manually trigger bower. Why doesnt this work via npm install?~
19 ADD .bowerrc /app/.bowerrc~
20 ADD bower.json /app/bower.json~
21 RUN bower install --config.interactive=false --allow-root~
22 ~
23 # Make everything available for start~
24 ADD . /app~
25 ~
26 ENV NODE_ENV production~
27 ~
28 # Port 3000 for server~
29 # Port 35729 for livereload~
30 EXPOSE 3000 35729~
31 CMD ["grunt"]~
```

# Assessment of user's code

## Example of Dockerfile

- Dockerfiles describe how to set up the application

```
Dockerfile
1 FROM node:latest
2
3 MAINTAINER Matthieu Nicolas, matthieu.nicolas@inria.fr
4
5 WORKDIR /app
6
7 # Required for PhantomJS
8 RUN apt-get install -y bzip2
9
10 # Install Mean.JS Prerequisites
11 RUN npm install -g grunt-cli
12 RUN npm install -g bower
13
14 # Install Mean.JS packages
15 ADD package.json /app/package.json
16 RUN npm install
17
18 # Manually trigger bower. Why doesnt this work via npm install?
19 ADD .bowerrc /app/.bowerrc
20 ADD bower.json /app/bower.json
21 RUN bower install --config.interactive=false --allow-root
22
23 # Make everything available for start
24 ADD . /app
25
26 ENV NODE_ENV production
27
28 # Port 3000 for server
29 # Port 35729 for livereload
30 EXPOSE 3000 35729
31 CMD ["grunt"]
```

- Run *docker build -t tag /path/to/Dockerfile* to build the image
- Start containers with *docker run tag*



# Assessment of user's code

More about *docker run*

- Can also manage
  - Ports

# Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes

# Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers

# Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers
  - Environment variables
  - Runtime constraints on resources
  - Restart policies
  - And a **lot more**

# Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers
  - Environment variables
  - Runtime constraints on resources
  - Restart policies
  - And a **lot more**
- Commands can become quite complex

```
docker run -p 443:9443 -link plm-accounts:accounts -v  
~/webPLM/logs/:/app/webplm-dist/logs webPLM
```

# Assessment of user's code

## Docker-compose

- Tool to easily deploy multi-containers applications

```
docker-compose.yml
1  nginx:
2    image: nginx
3    ports:
4      - "80:80"
5
6  messagequeue:
7    image: rabbitmq:3-management
8    ports:
9      - "5672:5672"
10     - "15672:15672"
11
12  plm:
13    image: webplm
14    volumes:
15      - ~/.plm:/root/.plm
16      - ~/webPLM/logs:/app/webplm-dist/logs
17    ports:
18      - "443:9443"
19    environment:
20      GITHUB_CLIENT_SECRET:
21      GOOGLE_CLIENT_SECRET:
22      PLMACCOUNTS_CLIENT_SECRET:
23      GITHUB_ACCESS_TOKEN:
24    links:
25      - accounts
26      - profiles
27      - messagequeue:messageq
28
29  accounts:
30    image: plm-accounts
31
32  profiles:
33    image: plm-profiles
34
35  db:
36    image: mongo
37    volumes:
38      - ~/mongodb/data:/data/db
39    ports:
40      - "27017:27017"
41
42  links:
43    - accounts
44    - profiles
45    - messagequeue:messageq
46
47  accounts:
48    image: plm-accounts
49    ports:
50      - "9000:3000"
51    links:
52      - db
53
54  profiles:
55    image: plm-profiles
56    ports:
57      - "8080:3000"
58    links:
59      - db
60
61  db:
62    image: mongo
63    volumes:
64      - ~/mongodb/data:/data/db
65    ports:
66      - "27017:27017"
```

- Deploy environment with *docker-compose up*

```

1  nginx:~
2    --image: nginx
3    --ports:~
4    --- - "80:80"~
5    ~
6  messagequeue:~
7    --image: rabbitmq:3-management~
8    --ports:~
9    --- - "5672:5672"~
10   --- - "15672:15672"~
11   ~
12  plm:~
13    --image: webplm~
14    --volumes:~
15    --- - "~/.plm:/root/.plm"~
16    --- - "~/webPLM/logs:/app/webplm-dist/logs"~
17    --ports:~
18    --- - "443:9443"~
19    --environment:~
20    --- GITHUB_CLIENT_SECRET:~
21    --- GOOGLE_CLIENT_SECRET:~
22    --- PLMACCOUNTS_CLIENT_SECRET:~
23    --- GITHUB_ACCESS_TOKEN:~
24    --links:~
25    --- - accounts
26    --- - profiles
27    --- - messagequeue:messageq~
28    ~
29  accounts:~
30    --image: plm-accounts~

```

```

20    --- GITHUB_CLIENT_SECRET:~
21    --- GOOGLE_CLIENT_SECRET:~
22    --- PLMACCOUNTS_CLIENT_SECRET:~
23    --- GITHUB_ACCESS_TOKEN:~
24    --links:~
25    --- - accounts
26    --- - profiles
27    --- - messagequeue:messageq~
28    ~
29  accounts:~
30    --image: plm-accounts~
31    --ports:~
32    --- - "9000:3000"~
33    --links:~
34    --- - db~
35    ~
36  profiles:~
37    --image: plm-profiles~
38    --ports:~
39    --- - "8080:3000"~
40    --links:~
41    --- - db~
42    ~
43  db:~
44    --image: mongo~
45    --volumes:~
46    --- - "~/mongodb/data:/data/db"~
47    --ports:~
48    --- - "27017:27017"~
49    ~

```

# Assessment of user's code

## Docker-compose

- Tool to easily deploy multi-containers applications

```
docker-compose.yml
1  nginx:
2    image: nginx
3    ports:
4      - "80:80"
5
6  messagequeue:
7    image: rabbitmq:3-management
8    ports:
9      - "5672:5672"
10     - "15672:15672"
11
12  plm:
13    image: webplm
14    volumes:
15      - ~/.plm:/root/.plm
16      - ~/webPLM/logs:/app/webplm-dist/logs
17    ports:
18      - "443:9443"
19    environment:
20      GITHUB_CLIENT_SECRET:
21      GOOGLE_CLIENT_SECRET:
22      PLMACCOUNTS_CLIENT_SECRET:
23      GITHUB_ACCESS_TOKEN:
24    links:
25      - accounts
26      - profiles
27      - messagequeue:messageq
28
29  accounts:
30    image: plm-accounts
31
32  profiles:
33    image: plm-profiles
34
35  db:
36    image: mongo
37    volumes:
38      - ~/mongodb/data:/data/db
39    ports:
40      - "27017:27017"
41
42  links:
43    - accounts
44    - profiles
45    - messagequeue:messageq
46
47  accounts:
48    image: plm-accounts
49    ports:
50      - "9000:3000"
51    links:
52      - db
53
54  profiles:
55    image: plm-profiles
56    ports:
57      - "8080:3000"
58    links:
59      - db
60
61  db:
62    image: mongo
63    volumes:
64      - ~/mongodb/data:/data/db
65    ports:
66      - "27017:27017"
```

- Deploy environment with *docker-compose up*



# Assessment of user's code

Docker in our case

- Deploy easily all components
- Restart judges automatically
- Limit users' mischiefs

## 1 Presentation of PLM

- Purposes
- Demo
- About PLM
- Architecture

## 2 Assessment of user's code

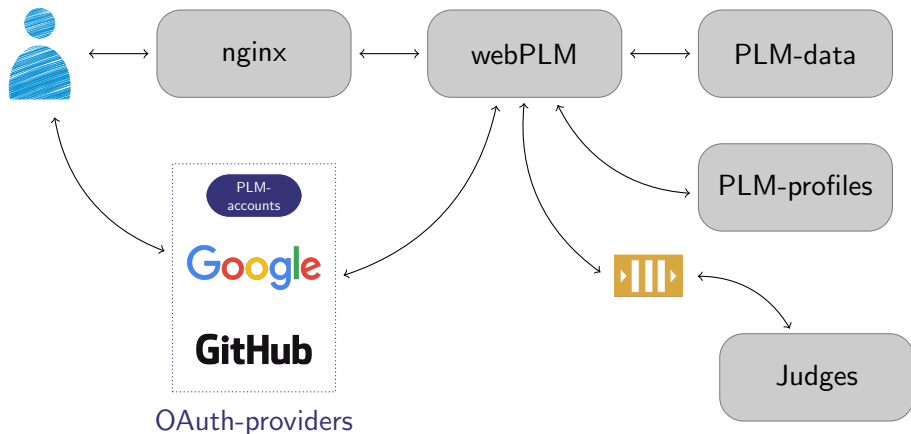
- Challenges
- Extraction of the execution component
- Docker

## 3 Result

## 4 Next steps

# Result

## Current architecture



# Result

Live-session in TELECOM Nancy

- Used in TELECOM Nancy in September 2015
- 30 hours of live testing with 100 students

# Result

Live-session in TELECOM Nancy

- Used in TELECOM Nancy in September 2015
- 30 hours of live testing with 100 students
- Engine is (almost) working fine...
- ... but user experience needs to be improved!

- Scalability issues:
  - Work well with small exercises
  - Can't cope with workload of larger exercises

# Result

Live-session in TELECOM Nancy

- Scalability issues:
  - Work well with small exercises
  - Can't cope with workload of larger exercises
- No tools for monitoring set up...

# Result

Live-session in TELECOM Nancy

- Scalability issues:
  - Work well with small exercises
  - Can't cope with workload of larger exercises
- No tools for monitoring set up...
- ... so the bottleneck is unknown.



## 1 Presentation of PLM

- Purposes
- Demo
- About PLM
- Architecture

## 2 Assessment of user's code

- Challenges
- Extraction of the execution component
- Docker

## 3 Result

## 4 Next steps

# Next steps

## Refactor the code

- Rushed to release a stable version before September 2015...
- Needed to clean some parts of the code
- Merged local and centralized mode branches

# Next steps

Simplify workflow to adapt the content

- Store most of content inside PLM
- Heavy and error prone workflow
- Need to extract the content from PLM's jar
- Allow to implement an exercise editor

# Next steps

Solve performance issues

- Set up some monitoring tools
- Perform some load testing to identify the bottleneck

Thanks for your attention, any questions?